VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

FITNESS APLIKACE PRO ANDROID

FITNESS APPLICATION FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Martin Husa
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Roupec, Ph.D.
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Martin Husa**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**Ing. Jan Roupec, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Fitness aplikace pro Android

Stručná charakteristika problematiky úkolu:

Mobilní telefon bude sloužit jako inteligentní plánovací tréninkový diář. Podle požadavků uživatele tento bude určovat tréninkové jednotky s návrhy jednotlivých cviků včetně ukázkových videí nebo obrázků, popisků a odpočtu času mezi sériemi.

Cíle bakalářské práce:

Zhodnocení situace v oblasti sportovních aplikací pro mobilní zařízení, stručný přehled vlastností platformy Android a možností vývoje aplikací pro tuto platformu. Analýza potřeb funkcí sportovní aplikace a návrh, realizace a otestování této aplikace.

Seznam doporučené literatury:

LACKO, Luboslav. Vývoj aplikací pro Android. Computer Press, Brno, 2015.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato bakalářská práce obsahuje rešeršní studii operačního systému Android, obecné základy programování aplikací pro Android a praktický příklad mobilní sportovní aplikace pro Android. Aplikace obsahuje rozpis tréninků doplněný o animace cviků a také databáze se statistikami z jednotlivých tréninků nebo měření váhy.

ABSTRACT

This Bachelor thesis contains search study about operating system Android, universally basics of application programming and practical example of mobile sport application for Android. Application contains training roster completed with animations of exercises and also databases with statistics of trainings or weight measurements.

KLÍČOVÁ SLOVA

Android, Software, Sport, Fitness, Kalistenika, Google, Android Studio, Java, Sportovní aplikace

KEYWORDS

Android, Software, Sport, Fitness, Calisthenics, Google, Android Studio, Java, Sport application

BIBLIOGRAFICKÁ CITACE

HUSA, Martin. *Sportovní aplikace pro Android*, Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky.

PODĚKOVÁNÍ

V první řadě bych chtěl poděkovat sportovcům Pavlu Beranovi a Ladislavu Přidalovi za poskytnutí informací o jejich tréninkových jednotkách a obrázků ze svých sociálních sítí, které jsou použity v aplikaci, jenž tato práce popisuje.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Jana Roupce, Ph.D. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 1. 3. 2017 ……

Martin Husa

OBSAH

1 ÚVOD

Smyslem práce je vývoj sportovní aplikace pro operační systém Android. Aplikace primárně slouží jako rozpis tréninků podle výběru s následnými animacemi cviků a výpočtem zkušenostních bodů podle splněných opakování jednotlivých cviků. Obsahuje jak tréninky se závažím, tak tréninky jenom s vlastní vahou, tzv. kalistenika, nebo tréninky cílené na jednotlivé svalové partie např. biceps. Obsahuje také tréninky podle primárních potřeb uživatele, např. tréninky na svalový objem nebo na dynamiku.

Aplikace mimo jiné také slouží jako diář pro zápis aktuální tělesné váhy uživatele s daným datem vážení. Uživatel tak má přehled o jeho dlouhodobém stavu tělesné váhy. Toto má velké uplatnění jak pro uživatele, kteří se snaží o redukci tělesné váhy, tak pro uživatele usilující o nabrání svalové hmoty.

V současné době se na trhu mobilních aplikací pro Android (Google Play) i pro jeho největšího rivala iOS (Apple Store) vyskytuje obrovské množství aplikací zahrnující naprostou většinu všech světových sportů. Existují aplikace týkající se přímo sportu např. aplikace měřící ujetou vzdálenost na kole, aplikace pro cvičení jógy, aplikace pro zlepšení fotbalové techniky s míčem, aplikace řešící stravu sportovce a pitný režim, což je nedílná součást sportovního výkonu.

Sportovní aplikace, kterou tato práce popisuje, je programována v operačním systému Android hlavně z důvodu větší otevřenosti pro vývojáře, než je tomu u konkurence. Důvodem použití operačního systému Android je i zdaleka největší zastoupení mezi mobilními zařízeními. Právě použití pro mobilní zařízení je u sportovních aplikací klíčové. Uživatel bude aplikaci používat během tréninku, tudíž použití na stolní PC by postrádalo význam.

Práce popisuje základní principy programování aplikací pro operační systém Android. Aplikace je vyvinuta v programovacím prostředí Android Studio od společnosti Google. Programovací prostředí obsahuje editor vzhledu jednotlivých aktivit, do kterých se umísťují **`widgety`** podle potřeby. Android Studio umožnuje editovat i soubory s Java kódy, ve kterých se jednotlivé prvky inicializují, propojují a oživují. Velmi podstatnou kladnou součástí Android Studia je i zahrnutí emulátoru, což je zařízení, které simuluje reálné mobilní zařízení.

V této práci jsou použity a vysvětleny jak základní principy architektury programu, tak další principy nutné pro chod aplikace jako `SharedPreferences`, `SQLite` databáze, `Intent` atd.

Vzhledem ke skutečnostem, že se jedná o nové a rychle rozvíjející se odvětví, nebyla ještě pro popis určitých prvků ustálena česká terminologie, proto jsou v této práci použity slangové termíny např. **`layout`** a **`widget`**. V příloze, jsou pak uvedeny vysvětlivky jednotlivých termínů.

2 ANDROID

Celá tato kapitola, popisující historii a architekturu operačního systému Android, je zpracována podle zdroje [1].

Android je open-source platforma na bázi operačního systému Linux určená hlavně pro mobilní zařízení. Systém Android vyvíjí organizace Open Handset Alliance, jejíž součástí jsou desítky firem včetně těch nejznámějších v odvětví mobilních zařízení např. Google, HTC, Intel, NVIDIA, Samsung.

Jde o jeden z mála operačních systémů, které podporují více platforem, které lze vidět u zařízení nejrůznějších značek. To však přináší jednu značnou nevýhodu – **chybí optimalizace systému pro konkrétní platformu**, což je silná zbraň Apple iOS. Největší výhodou a zároveň nevýhodou platformy je její otevřenost a možnost úprav, ať už ze strany výrobců nebo uživatelů. Úpravy se netýkají jen konfigurace komponentu tzv. **`widgetu`**, ale i firmwaru.

Mobilní zařízení s Androidem vydává mnoho firem. Je to záruka dynamičtějšího vývoje nových zařízení, například v porovnání s Applem, kde je celý vývoj hardwaru v režii jedné firmy, a zároveň to představuje problém, protože aplikace běží na přístrojích s různým rozlišením displeje a různým výkonem procesoru a grafiky.

2.1 Historie Androidu a jeho verzí

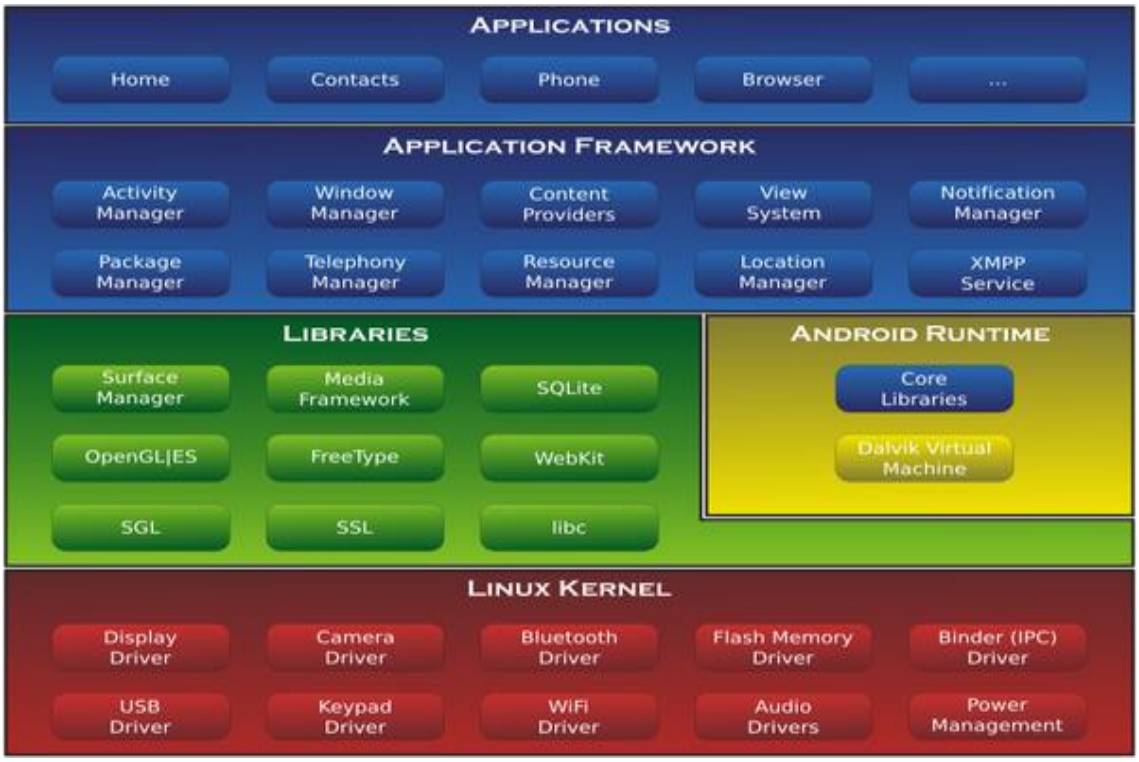
Android nebyl vyvinut ani navržen firmou Google, ale Google Android odkoupil. Společnost Android Inc. vznikla roku 2003. Založili ji čtyři zakladatelé v kalifornském městě Palo Alto. O dva roky později, v srpnu 2005, Google odkoupil Android Inc. za 50 milionů dolarů. Dnes má hodnotu vyšší o tři řády.

V listopadu 2007 byla založena Open Handset Alliance, která stojí za vývojem Androidu dodnes, v té době vyšel zároveň vývojářský kit (SDK). V září 2008 přišel v USA na trh HTC Dream (G1), první chytrý telefon s Androidem 1.0, který měl na trhu zanedbatelný podíl 0,5 %. V dubnu 2009 spatřil světlo světa první masový Android ve verzi 1.5 (Cupcake). Jednotlivé verze operačního systému Android (Obr. 1) mají kromě číselného označení i kódové – tím je název zákusku, v abecedním pořadí (Cupcake, Donut, Eclair atd.).

Obr. 1: Loga verzí Androidu [2]

2.2 Architektura operačního systému Android

Pro vývoj aplikací je nezbytné znát základní informace o principech a architektuře operačního systému Android, ve kterém budou aplikace spuštěny. Architektura Androidu je podle schématu (Obr. 2) popsána od nejnižší architektonické vrstvy po nejvyšší.



Obr. 2: Schéma architektury operačního systému Android [4]

2.2.1 Linux Kernel

Základním pilířem, nejnižší vrstvou architektury Androidu, je upravené jádro populárního operačního systému Linux. Úpravy se týkají redukce funkcí a jejich přizpůsobení možnostem mobilních zařízení.

Jádro slouží k přímé interakci s hardwarem mobilního zařízení, čímž zabezpečuje úplnou abstrakci od hardwaru pro vyšší softwarové vrstvy. Zabezpečuje správu paměti, správu procesů, základní síťovou vrstvu a ovladače. Řízení procesů umožňuje, aby více procesů běželo současně, aniž by se vzájemně ovlivňovaly.

Na úrovni jádra je implementované i zabezpečení systému, správa napájení, vstupně-výstupní operace či základní grafika. Podle hardwarové konfigurace jsou na této úrovni i moduly ovladačů pro Bluetooth, EDGE, 3G, Wi-Fi, fotoaparát, GPS.

2.2.2 Libraries

Nad jádrem je situovaná vrstva knihoven, které poskytují přímý přístup aplikací k různým komponentám systému Android. Jsou to nativní knihovny napsané v C/C++. Tvoří mezivrstvu mezi různými komponentami vyšších vrstev a linuxovým jádrem.

Modul Surface Manager podporuje funkcionalitu multidotykového displeje. Zabezpečuje výslednou kompozici grafického výstupu více aplikací do souvislého toku dat, který směřuje do grafické vyrovnávací paměti. Z této paměti probíhá vykreslování na obrazovku.

Webkit je určený k renderování a zobrazení webových stránek. Na této úrovni jsou implementované knihovny medií, grafické 2D a 3D knihovny, přičemž 3D knihovny využívají podporu OpenGL ES (OpenGL for Embedded Systems) s volitelnou možností využití grafických akcelerátorů.

Aplikace pracující s daty mohou využít `SQLite`. Knihovny v této vrstvě suplují linuxové funkce, o které bylo jádro operačního systému redukováno.

2.2.3 Android Runtime

Android Runtime obsahuje sadu základních knihoven. Každá aplikace pro Android je samostatný proces využívající vlastní instanci virtuálního stroje DVM (Dalvik Virtual Machine). Tento zabezpečuje běh spustitelných souborů s příponou DEX. Soubory DEX vznikly kompilací z klasických souborů CLASS a JAR. Dalvik je optimalizovaný pro mobilní zařízení. To znamená, že bere v úvahu omezení možnosti napájení, menší paměť a podobně.

2.2.4 Application Framework

Aplikační framework obsahuje v aplikacích opakovaně používaný software, například ovládací prvky, ikony a podobně. Poskytuje aplikacím základní služby systému.

Package Manager - modul správce balíčků je v podstatě databáze, která udržuje aktuální seznam všech aplikací nainstalovaných v zařízení.

Window Manager - spravuje okna, která tvoří aplikace. Aplikace využívají většinou dvě a více oken současně.

View System - spravuje širokou paletu společných prvků grafického uživatelského rozhraní, jako jsou ikony, tlačítka, prvky na zobrazení a editování textu a mnohé další.

2.2.5 Aplikace

Nejvyšší úrovní operačního systému Android jsou aplikace, například program na posílání zpráv, navigaci, kalendář a podobně.

2.3 Základní součásti aplikace pro Android

Aplikace pro Android jsou vybudované na čtyřech základních pilířích realizovaných jako třídy.

2.3.1 Aktivity

Aktivita je hlavní třída, která se uživateli zobrazí po spuštění aplikace. Aktivity umožňují uživatelům přes grafické rozhraní (GUI) přijímat informace od aplikace a ovládat ji. Třída aktivity je předurčena k tomu, aby zobrazovalo uživatelské rozhraní a zachytávala interakce uživatele přes toto rozhraní.

2.3.2 Služby

Služby realizují déle trvající operace na pozadí. Na rozdíl od aktivity běží služby na pozadí a nepotřebují uživatelské rozhraní. Služby umožňují asynchronně, paralelně s hlavním vláknem, provádět operace, jejichž realizace trvá déle. Typickým příkladem služby je přehrávání hudby na pozadí.

2.3.3 Broadcast Receivers

Objekty na vysílání a přijímání poslouchají na pozadí a reagují na události, které se odehrávají na zařízení. Události jsou zastoupeny objekty typu: `Intent` (záměr). Vydavatelé vytvářejí záměry a následně je přes Broadcast směřují do vysílání. Zachytávají je přijímače, které mají příslušné záměry objednané nebo registrované. Dobrým příkladem je přijetí SMS zprávy nebo emailu.

2.3.4 Poskytovatelé obsahu

Poskytovatelé obsahu umožňují ukládání a sdílení dat mezi více aplikacemi a procesy. Aplikace tak mohou přistupovat k údajům ostatních aplikací, které vystupují jako poskytovatelé obsahu. Využívá se rozhraní podobné databázovému, poskytovatelé obsahu jsou ale více než jen databáze.

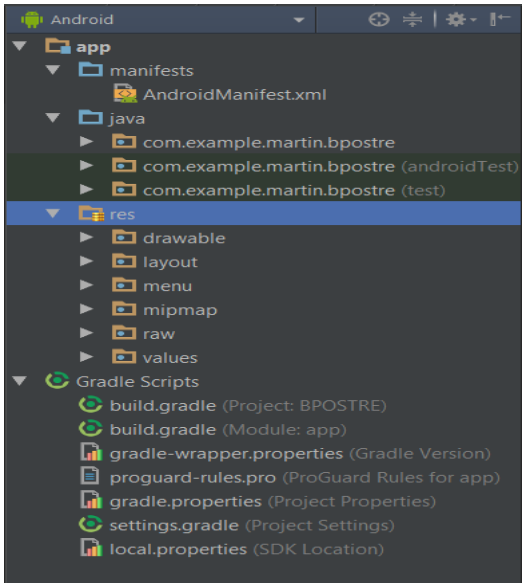
3 ANDROID STUDIO (VÝVOJOVÉ PROSTŘEDÍ)

Android Studio je nové vývojové prostředí založené na IntelliJ IDEA. Android studio bylo firmou Google oficiálně představeno 16. května 2013 na konferenci Google I/O. Od června 2013 je zdarma k dispozici pro uživatele na platformách Windows, Mac OS X a Linux. Před uvedením Android Studia bylo základním, doporučeným, způsobem vývoje Android aplikací prostředí Eclipse s pluginem Android Developer Tools. [3]

Obr. 3: Logo programu Android Studio [5]

3.1 Anatomie projektu

Každý projekt v Android Studio má anatomii (Obr. 4), kterou tvoří 4 základní složky.



Obr. 4: Anatomie projektu v Android Studio

Manifest

Manifest je hlavní konfigurační soubor aplikace. Definuje jednotlivé komponenty, nastavení konfigurace a oprávnění aplikace.

Java

Obsahuje soubory s Java kódy ke všem aktivitám aplikace, databázím a podobně. Kódy je zde možné i editovat.

Res

Ve složce res se nacházení XML soubory s definicí uživatelského rozhraní např. v podsložce **`drawable`** se nacházejí importované prvky typu **`backgroud`**, podsložka **`layout`** zase obsahuje celistvé vzhledy jednotlivých aktivit.

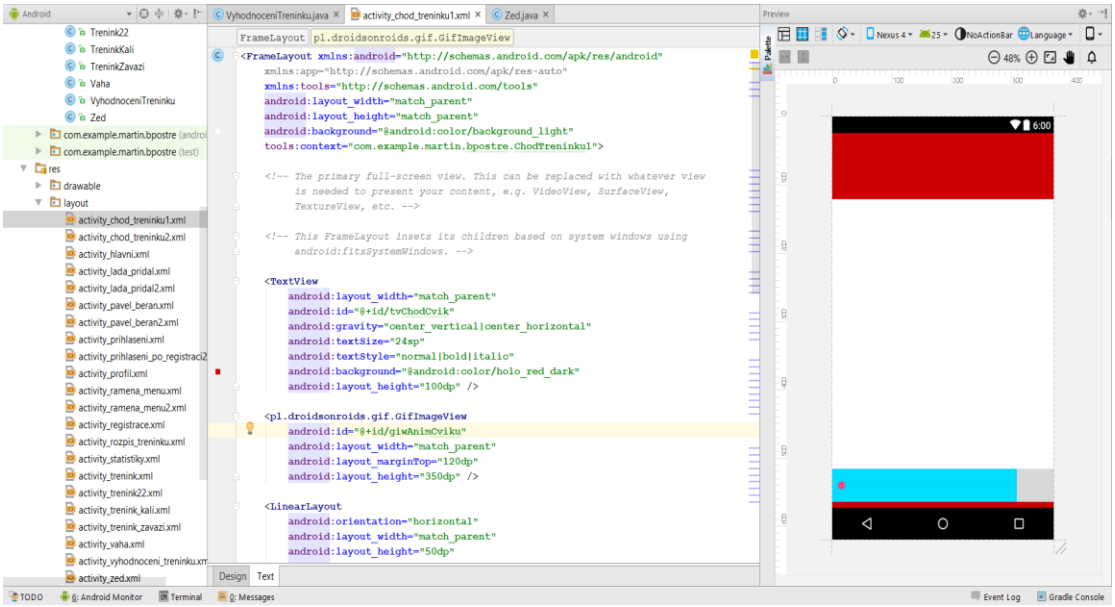
Gradle Scripts

Android Studio je spjaté s nástrojem **`Gradle`**. Zde se v podsložce **`build.gradle`** přidávají části kódu, které implementují další funkce aplikace např. požití grafů nebo animovaných gifů.

3.2 Design Aktivit

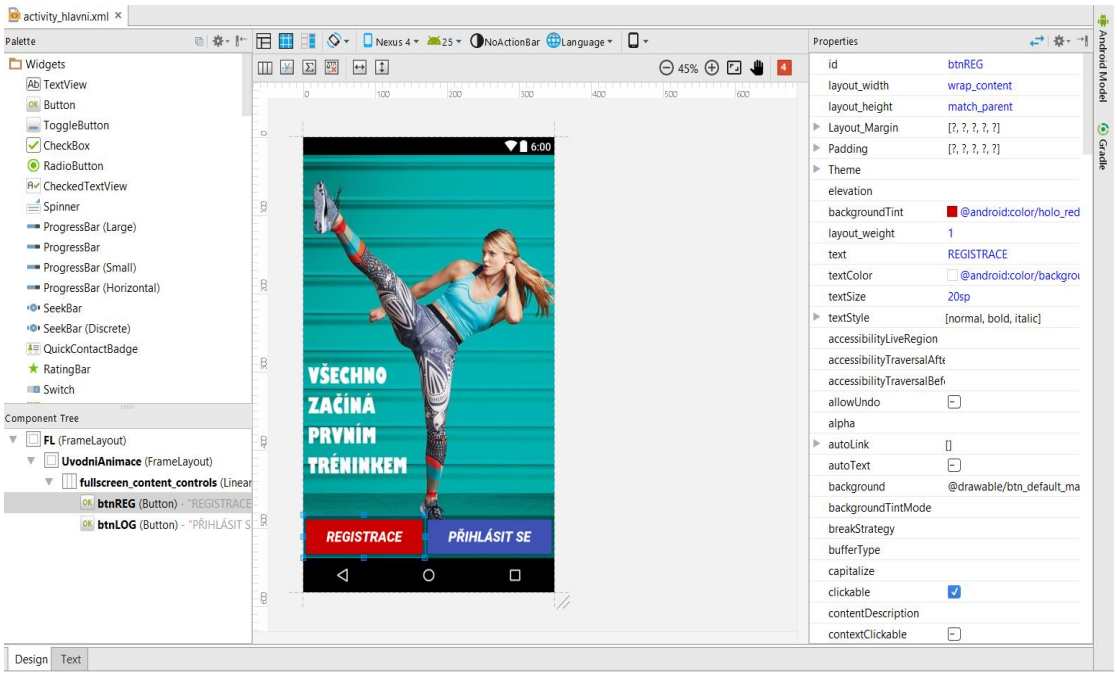
Android Studio umožňuje uživatelům dvě metody tvorby vzhledu aplikací. Stejně jako u předchůdců Android Studia je možné design vytvářet ve dvou módech – XML mód a Design mód.

Při psaní v XML módu (Obr. 5), program zobrazuje náhled na aktuálně vytvářený design. Všechny vzhledové vlastnosti jednotlivých **`widgetů`** jsou nastavovány právě přes XML kód.



Obr. 5: Design aktivity v XML módu

Při vytváření designu prostřednictvím Design módu (Obr. 6) se ve středu obrazovky nachází **`layout`** aktivity. V levé části obrazovky je seznam všech **`widgetů`** s názvem **`Palette. Widgety`** lze jen lehkým přetáhnutím umístit na **`layout`** aktivity, nacházející se ve středu obrazovky. Pod seznamem **`Pallette`** je seznam již všech použitých **`widgetů Component Tree.`** V pravé části obrazovky je pak seznam **`Properties`**, se zobrazují vlastnosti vybraného **`widgetu`**.



Obr. 6: Design aktivity v Design módu

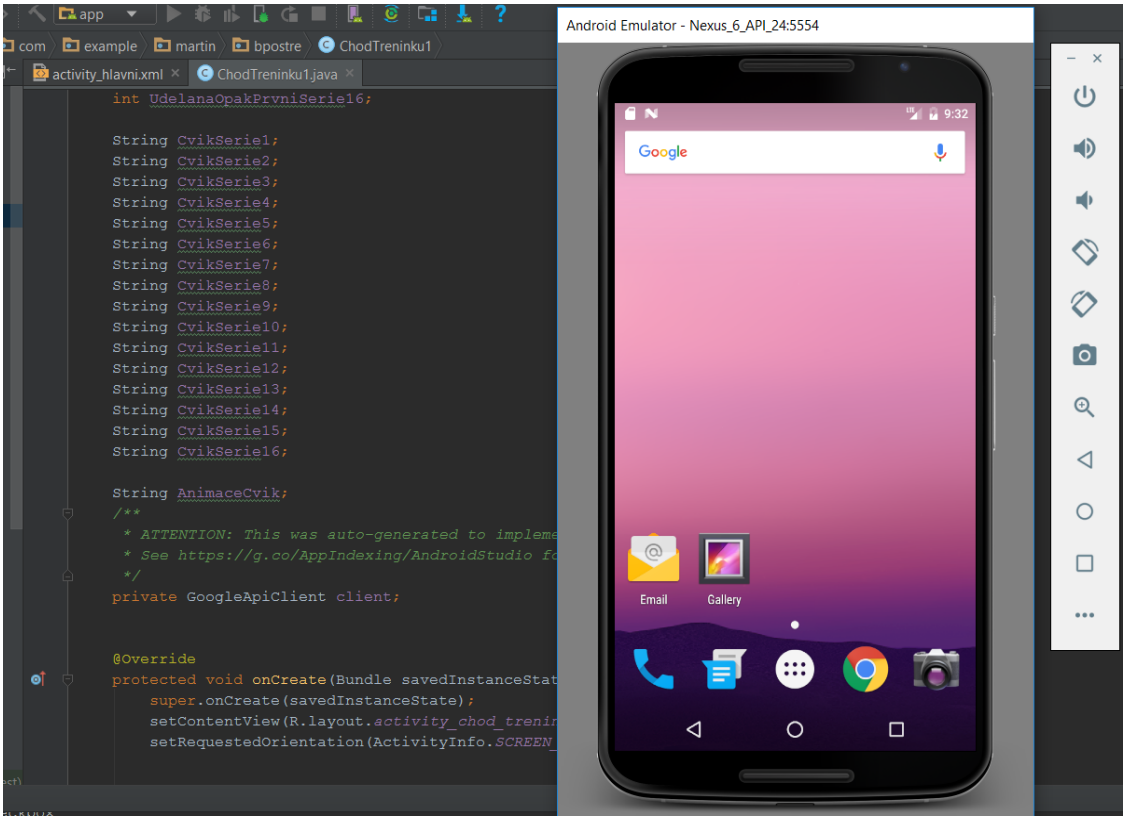
3.3 Zdrojový kód – Java

Zde se editují soubory s Java kódy, ve kterých se jednotlivé **`layouty`** a **`widgety`** inicializují, propojují a doslova oživují. Vzniká zde tedy kompletní logika programu aplikace. Android studio obsahuje i tzv. našeptávač, který usnadňuje psaní Java kódu.

3.4 Emulátor

Android Studio umožňuje spustit projekt na zařízení, které simuluje reálné mobilní zařízení, na tzv. emulátoru. Do Android Studia je možné dodatečné stáhnout velké množství emulátorů podle potřeb uživatele, jde zejména o rozměr displeje.

Android studio umožňuje po propojení telefonu s PC pomocí USB kabelu a po povolení vývojářských možností a ladění USB na vašem mobilním telefonu nahradit emulátor přímo mobilním telefonem. Je možné tak aplikaci testovat přímo na reálném zařízení.



Obr. 7: Spouštění emulátoru: Nexus 6

4 PROGRAM

Program této aplikace je napsán v Java kódu ve vývojovém prostředí Android Studio. Aplikace obsahuje dva styly tréninku: trénink se závažím nebo s vlastní vahou tzv. kalistenika. Při volbě tréninku se závažím aplikace umožňuje uživateli trénink cílený na jednotlivé svalové partie např. hrudník. Dále si uživatel volí své požadavky od tréninku např. svalový objem nebo dynamika a tři stupně náročnosti tréninku. Na výběr jsou i předvolené tréninkové sety od dvou představitelů jak pro kalisteniku, tak pro trénink se závažím.

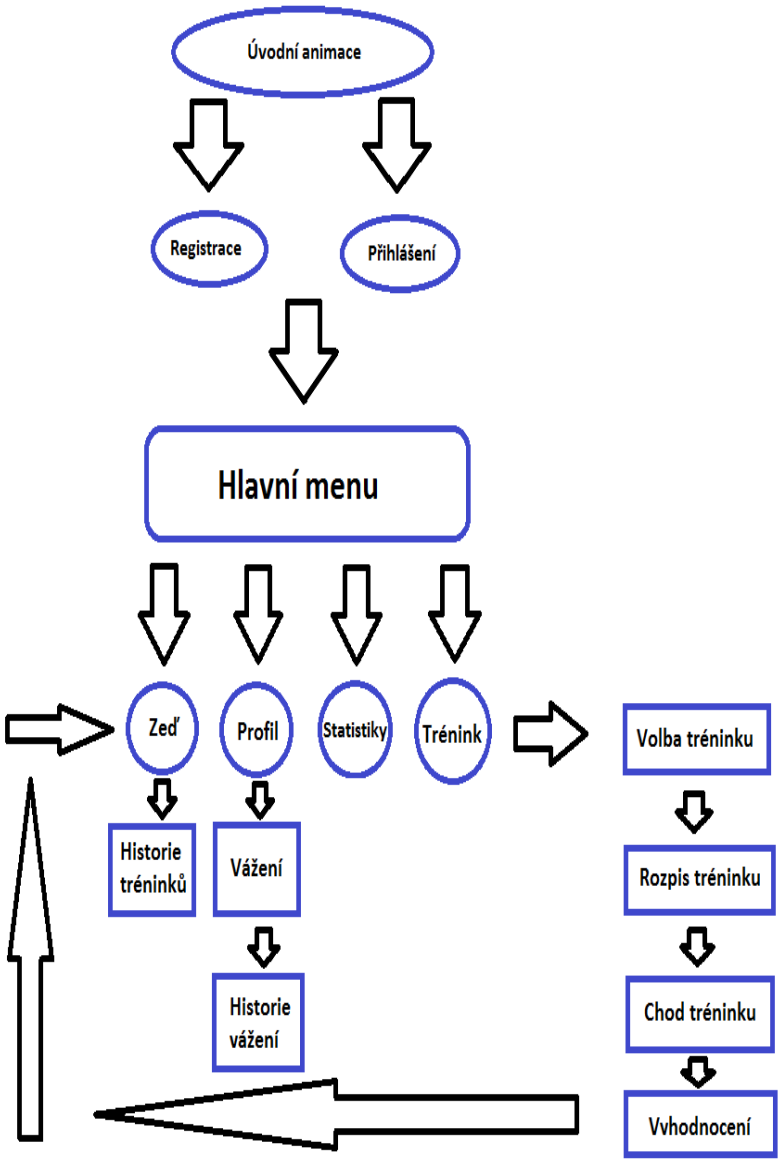
Uživatel u každého cviku zadává splněná opakování a program vypočítává zkušenostní body, které se liší i podle náročnosti cviku nebo tréninku. Následně jsou celkové zkušenostní body přiřazeny k aktuálnímu datu a uloženy do databáze.

Je zde obsažen i diář pro zápis jednotlivých dat s aktuální tělesnou váhou uživatele a grafickým zobrazením. Uživatel tedy má názorný přehled o jeho vývoji tělesné váhy.

4.1 Analýza a návrh aplikace

Aplikace obsahuje následující aktivity:

- Úvodní animace dvěma tlačítky: aktivita pro přihlášení a pro registraci
- Hlavní menu (nabídka)
- Zeď - Historie vážení
- Profil - Vážení + Historie vážení
- Statistiky
- Trénink
  - Volba tréninku
  - Rozpis tréninku
  - Chod tréninku
  - Vyhodnocení tréninku



Obr. 8: Diagram aktivit

4.2 Úvodní animace a Intent

Vzhled úvodní aktivity (Obr. 9) se skládá z prvku **`FrameLayout`**, do kterého je promítaná animace, a dvou tlačítek – přihlášení a registrace.

Obr. 9: Úvodní aktivita

4.2.1 Animace

Animace je vytvořena pomocí prvku **`AnimationDrawable`** (Obr. 10). Nejdříve byl tedy vytvořen prvek **`AnimationDrawable`**, následně inicializován **`FrameLayout`**. Do animace byly pak vloženy tři obrázky ze složky **`drawable`** a nastavena délka animace každého obrázku na 3000ms. Potom byla animace vložena do **`FrameLayoutu`** a spuštěna.

```
AnimationDrawable animace = new AnimationDrawable();
FrameLayout Napis = (FrameLayout) findViewById(R.id.UvodniAnimace);

animace.addFrame(getResources().getDrawable(R.drawable.uvodnianim), 3000);
animace.addFrame(getResources().getDrawable(R.drawable.cinka2), 3000);
animace.addFrame(getResources().getDrawable(R.drawable.calli2), 3000);


Napis.setBackground(animace);

animace.start();
```

Obr. 10: Java kód úvodní animace

4.2.2 Intent

Pro přeskočení na další aktivitu byl použit prvek `Intent`. Nejdříve bylo inicializováno tlačítko registrace a poté na něj použita funkce *setOnClickListener()*, do které byl přidán nový `intent` s odkazem na další aktivitu: **Registrace**. Obdobně bylo řešeno i přeskočení na aktivitu **Přihlášení**.

`Intent` je asynchronní zpráva, která nese informaci o požadované akci. Akcí může být spuštění aktivity, služby, případně uživatelsky definované akce. `Intent` definuje vstupy a výstupy aktivit a umožňuje vzájemnou komunikaci volně vázaných komponent.[1]

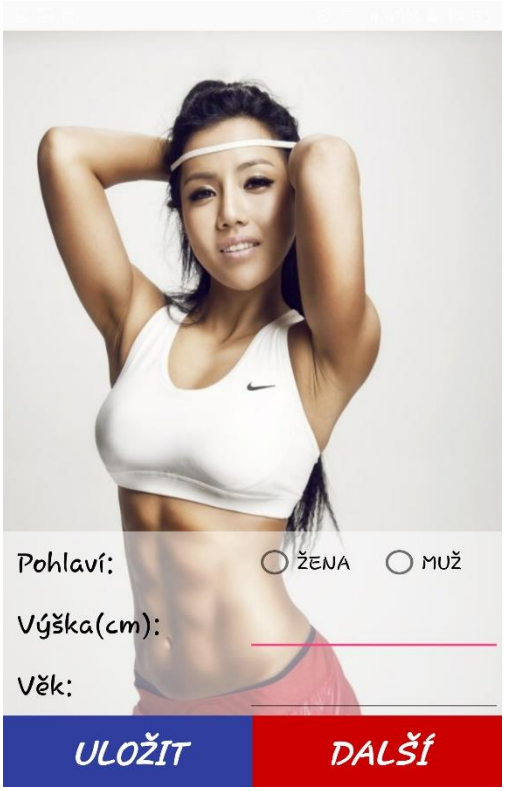
```
public void OnClickButtonListener()
{
    BTNreg = (Button) findViewById(R.id.btnREG);
    BTNreg.setOnClickListener(
            new View.OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    Intent intent = new Intent("com.example.martin.bpostre.registrace");

                    startActivity(intent);
                }
            }
    );

}
```

Obr. 11: Java kód přeskočení na aktivitu Registrace

4.3 Aktivity Registrace, Přihlášení a SharedPreferences

4.3.1 Aktivita Registrace

Aktivita (Obr.12) je tvořena prvkem **`FrameLayout`**, do kterého se vykresluje pozadí aktivity, několika **`ImageView`** a **`EditTexty`** pro zadání vstupních veličin uživatele, je zde i několik tlačítek pro uložení i pro přeskočení na další aktivitu. V designu registrace jsou použity dva **`RadioButtony`** pro volbu pohlaví. **`RadioButtony`** jsou uloženy do **`RadioGroup`**, což způsobí, že lze zvolit pouze jeden z nich.



Obr. 12: Aktivita Registrace

Vzhledem ke skutečnostem, že uživatel může zadat nesmyslné údaje např. výška 999cm, musely být vytvořeny dvě kontrolní funkce: *KontrolaUlozeni()*, *VyhodnoceniRegistrace()* (Obr. 13).

```
public void KontrolaUlozeni()
{
    ETVyska = (EditText) findViewById(R.id.editTextVyska);
    ETRok = (EditText) findViewById(R.id.editTextNarozeni);

    Vyska2 = ETVyska.getText().toString();
    Rok2 = ETRok.getText().toString();

    SharedPreferences sp33 = getSharedPreferences("FyzickeUdaje", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor33 = sp33.edit();

    editor33.putString("Pohlavi", Pohlavi);
    editor33.commit();

    editor33.putString("Vyska2", Vyska2);
    editor33.commit();

    editor33.putString("Rok2", Rok2);
    editor33.commit();

    if(Pohlavi != "MUŽ" && Pohlavi != "ŽENA")
    {
        Toast.makeText(this, "Nezvolil jste pohlaví.", Toast.LENGTH_LONG).show();
    }

    if (Vyska2.length() == 0)
    {
        Toast.makeText(this, "Nezadal jste výšku.", Toast.LENGTH_LONG).show();
    }

    else
    {
        Vyska = Integer.parseInt(ETVyska.getText().toString());

        if ((Vyska >= 130) && (Vyska <= 230))
        {
            VyskaProsla = true;

        }
        else
        {

            VyskaProsla = false;

            Toast.makeText(this, "Vaše výška je zadaná špatně. Zadejte výšku v rozmezí od 130cm do 230cm.", Toast.LENGTH_LONG).show();

        }

    }

    if (Rok2.length() == 0)
    {
        Toast.makeText(this, "Nezadal jste rok.", Toast.LENGTH_LONG).show();

    } else
    {
        Rok = Integer.parseInt(ETRok.getText().toString());

        if ((Rok >= 10) && (Rok <= 70))
        {

            RokProsel = true;

        }
        else
        {

            RokProsel = false;
            Toast.makeText(this, "Váš věk je zadán špatně. Zadejte věk v rozmezí od 10 do 70.", Toast.LENGTH_LONG).show();

        }
    }
    if (RokProsel == true && VyskaProsla == true)
    {

        Toast.makeText(this, "Vaše údaje jsou uloženy.", Toast.LENGTH_LONG).show();
    }
}
```

Obr. 13: KontrolaUlozeni()

Funkce *KontrolaUlozeni()* nejdříve inicializuje **`EditTexty`** pro zadávání údajů a poté z nich načte jejich hodnoty. Hodnoty jsou následně uloženy do `SharedPreferences` a použity také na ošetření nereálných situací. Pro případ, že údaje jsou zadány správně, jsou zde dvě proměnné typu `boolean`: ***RokProsel*** a ***VyskaProsla***, které získají hodnotu true a jsou použity ve funkci *VyhodnoceniRegistrace()* (Obr. 14) jak pro přeskočení do další aktivity pomocí `Intentu`, tak pro error hlášení.

```
public void VyhodnoceniRegistrace()
{

    if (VyskaProsla == true&&  RokProsel == true)
    {

        Intent intent = new Intent("com.example.martin.bpostre.PrihlaseniPoRegistraci2");
        startActivity(intent);

    }

    else
    {

        Toast.makeText(this, "Zadejte a uložte svoje údaje.", Toast.LENGTH_LONG).show();


    }

}
```

Obr. 14: VyhodnoceniRegistrace()

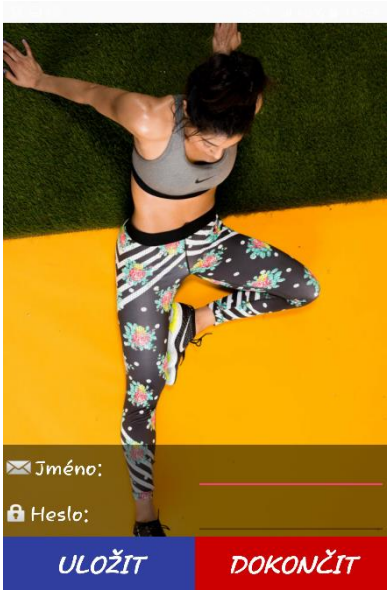
4.3.2 SharedPreferences

Třída `SharedPreferences` umožňuje aplikacím ukládání a správu malého množství jednoduše strukturovaných údajů. Je implementována jako perzistentní mapa k ukládání párových hodnot typu klíč-hodnota.

Údaje uložené přes `SharedPreferences` jsou perzistentní, takže údaje uložené během spuštění aplikace budou dostupné i po jejím ukončení a restartu.[1]

4.3.3 Aktivity Přihlášení po registraci, Přihlášení

Aktivita **Přihlášení po registraci** (Obr. 15) slouží pro zadání přihlašovacích údajů, jako jsou jméno a heslo uživatele. Pro uložení je opět použit prvek `SharedPreferences`, a je zde i ošetřena minimální i maximální délka jména či hesla na stejném principu jako u předešlé třídy.



Obr. 15: Aktivita Přihlášení po registraci

Aktivita **Přihlášení** (Obr. 16) si logicky od uživatele bere stejné údaje jako Aktivita **Přihlášení po registraci**. Funkce *VyhodnotPrihlaseni()* (Obr.17) si načte jméno a heslo uživatele ze `SharedPreferences`, které uživatel zadal v aktivitě **Přihlášení po registraci** a při shodě pomocí prvku `Intent` spouští hlavní menu jmenovitě aktivitu **Zeď**.

Obr. 16: Aktivita Přihlášení

```
public void VyhodnotPrihlaseni()
{

    ETJmeno = (EditText) findViewById(R.id.ETJmeno);
    Jmeno = ETJmeno.getText().toString();

    ETHeslo = (EditText) findViewById(R.id.ETheslo);
    Heslo = ETHeslo.getText().toString();

    SharedPreferences sp5 = getSharedPreferences("Registrace", Context.MODE_PRIVATE);

    JmenoSP = sp5.getString("Jmeno2", "");
    HesloSP = sp5.getString("Heslo2", "");

    if (Jmeno.equals(JmenoSP) && Heslo.equals(HesloSP))
    {
        Intent intent = new Intent("com.example.martin.bpostre.Zed");

        startActivity(intent);
    }

    else {
        Toast.makeText(this, "Vaše přihlašovací údaje nejsou správné.",Toast.LENGTH_LONG).show();

    }


}
```

Obr. 17: VyhodnotPrihlaseni()

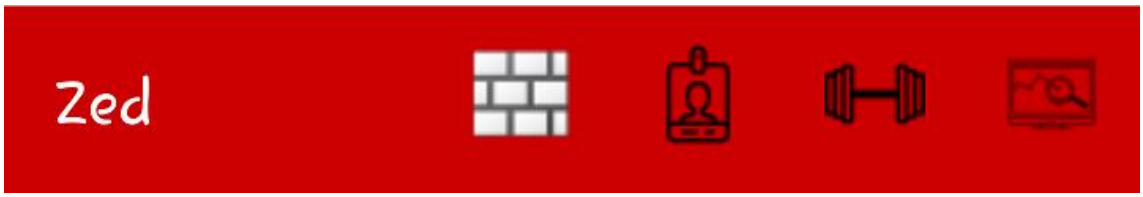
4.4 Hlavní menu, Action bar, SQLite databáze

Hlavní menu aplikace je tvořeno prvkem zvaný **`Action bar`** (Obr. 18), který slouží pro spouštění čtyř základních aktivit: **Zeď, Profil, Trénink** a **Statistiky**.

4.4.1 Action Bar

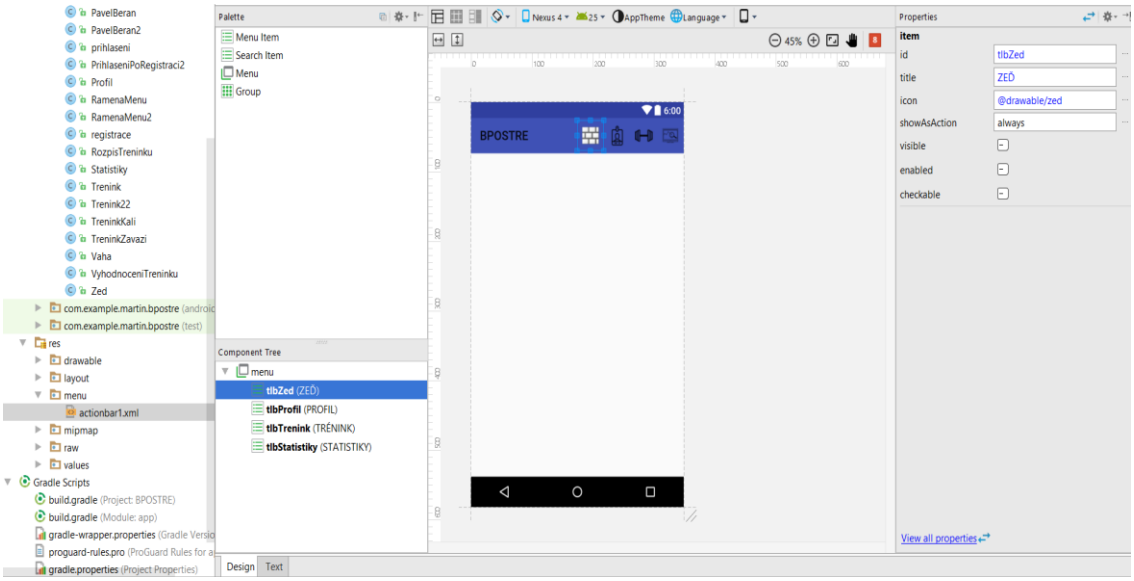
`Action Bar` je lišta s tlačítky v horní části každé obrazovky k aktivování nejpoužívanějších aktivit. Tato lišta je perzistentní napříč celým hlavním menu, takže umožňuje konzistentní navigaci a přepínání zobrazení v rámci aplikace.

`Action Bar` je pro uživatele jakýmsi záchytným bodem, standardizovaným prvkem umožňujícím rychle si osvojit ovládání aplikace. Operační systém navíc **`Action Bar`** elegantně přizpůsobí různým konfiguracím obrazovky.[1]



Obr. 18: Action Bar – Lišta

Složka res obsahuje podsložku menu (Obr. 19), která slouží k vytvoření designu prvku **`Action Bar`**. Jedná se o vzhled ikon jednotlivých aktivit, ale i o jejich pořadí v liště.



Obr. 19: Action Bar – menu

Tento kód (Obr. 20) s danou funkcí je nutné vložit, do každé dílčí aktivity, `kterou` **`Action Bar`** zahrnuje. Jde jednak o inicializaci a přeskok na další aktivity opět pomocí prvku `Intent`.

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    MenuInflater MI=getMenuInflater();
    MI.inflate(R.menu.actionbar1,menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    int I=item.getItemId();

    if(I==R.id.tlbZed)
    {
        Intent intent = new Intent("com.example.martin.bpostre.Zed");
        startActivity(intent);
    }
    else if (I==R.id.tlbTrenink)
    {
        Intent intent = new Intent("com.example.martin.bpostre.Trenink");
        startActivity(intent);
    }
    else if (I==R.id.tlbStatistiky)
    {
        Intent intent = new Intent("com.example.martin.bpostre.Statistiky");
        startActivity(intent);
    }
    else if (I==R.id.tlbProfil)
    {
        Intent intent = new Intent("com.example.martin.bpostre.Profil");
        startActivity(intent);
    }
    return super.onOptionsItemSelected(item);
}
```

Obr. 20: Action Bar – Java

Do Java kódu dílčích aktivit **`Action Baru`** (Obr. 21) je nutné také ve funkci *OnCreate()* **`Action Bar`** inicializovat s prvkem **`widget`**, který byl nastaven v design módu.

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_zed);
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_NOSENSOR);

    tb1 = (Toolbar) findViewById(R.id.TB1);
    setSupportActionBar(tb1);
```

Obr. 21: Action Bar – Java 2

4.4.2 Aktivita Zeď

Vzhled této aktivity (Obr. 22) je tvořen prvkem **`Action Bar`**, hned pod ní lišta, která zobrazí pozdrav uživateli. Pozdrav je uskutečněn opět přes prvek `SharedPreferences`. Jednoduše načte jméno uživatele, které zadal při registraci, a zobrazí ho do **`widgetu`**. Ve střední části, se nachází kalendář se zobrazeným aktuálním datem.

Ve spodní části je tlačítko k zobrazení historie tréninků. Historie tréninků je řešena přes databází `SQLite`. Databáze má vytvořenou vlastní aktivitu s názvem **DatabaseHelper3**. Tato aktivita je pouze pomocná a nemá vlastní **`layout`**.

Obr. 22: Aktivita Zeď

Pomocná aktivita **DatabseHelper3** (Obr. 23) slouží k nastavení jména databáze i jména tabulky a jejich jednotlivých sloupců. Do aktivity jsou naimplementované funkce pro vložení a zobrazení dat. Dále je zde funkce *Sum(),* která je v tomto případě používaná pro sčítání zkušenostních bodů získaných v jednotlivých trénincích.

```java
public class DatabaseHelper3 extends SQLiteOpenHelper
{
    public static final String DATABASE_JMENO = "Treninky.db";
    public static final String TABLE_JMENO = "Treninky_table";

    public static final String COL_1 = "ID";
    public static final String COL_2 = "DATUM";
    public static final String COL_3 = "XP";


    public DatabaseHelper3(Context context) {
        super(context, DATABASE_JMENO, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db)
    {
        db.execSQL("create table " + TABLE_JMENO + " (ID INTEGER PRIMARY KEY AUTOINCREMENT, DATUM INTEGER, XP INTEGER)");

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
    {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_JMENO);
        onCreate(db);
    }

    public boolean insertData(String datum, String xp)
    {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();

        contentValues.put(COL_2, datum);
        contentValues.put(COL_3, xp);

        long result = db.insert(TABLE_JMENO, null, contentValues);

        if (result == -1)
            return false;
        else
            return true;
    }

    public Cursor getAllData()
    {
        SQLiteDatabase db = this.getWritableDatabase();

        Cursor res = db.rawQuery("select * from " + TABLE_JMENO, null);
        return res;
    }

    public int Sum()
    {
        int total = 0;

        SQLiteDatabase dbb = this.getWritableDatabase();

        Cursor c = dbb.rawQuery("SELECT SUM(" + (COL_3) + ") FROM " + TABLE_JMENO, null);

        if (c.moveToFirst())
        {
            total = c.getInt(0);
        }
        return total;
    }
}
```

Obr. 23: DatabaseHelper3

V Java kódu aktivity Zeď (Obr. 24), v řídící funkci *OnCreate()* je nejdříve nutné databázi inicializovat.

```
mojeDB = new DatabaseHelper3(this);
```

Obr. 24: Inicializace SQLite databáze

Dále je nutné do aktivity přidat funkce pro přidání a zobrazení dat (Obr. 25). V této části kódu se nastavuje i formát zobrazovaní.

```
public void UkažVše()
{
    BTNHistorieTreninku.setOnClickListener(
            new View.OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    Cursor ress = mojeDB.getAllData();

                    if(ress.getCount() == 0)
                    {
                        UkažMessage("Error", "Data nenalezena");
                        return;
                    }

                    StringBuffer bufferr = new StringBuffer();
                    while (ress.moveToNext())
                    {
                        bufferr.append("Id :"+ ress.getString(0)+"\n");
                        bufferr.append("Datum :"+ ress.getString(1)+"\n");
                        bufferr.append("Xp :"+ ress.getString(2)+"\n\n");
                    }

                    UkažMessage("Data", bufferr.toString());
                }
            }
    );
}
public void PridejData()
{
    mojeDB.insertData(DatumTreninku.toString() +". "+ DatumTreninku2.toString()+". " + DatumTreninku3.toString()+". ", XXXP.toString() + "Xp");
}
```

Obr. 25: SQLite databáze - Java

4.4.3 Databáze SQLite

Embedded databáze `SQLite` (SQLite.org) má široké uplatnění na různých platformách, ať už je to Android, Windows 8, iOS, Mono nebo třída Python.

`SQLite` je malá, ale výkonná relační databáze, která pracuje bez serveru, nemusí se instalovat, nemá žádné konfigurační soubory Na rozdíl od SQL Serveru, Oraclu, MySQL a jiných řešení klient-server používá `SQLite` pouze klientskou část. Data jsou uložená v souboru, ke kterému přistupuje přímo klientská aplikace. Zjednodušeně řečeno samotná aplikace zastává funkci klienta i serveru současně.

I když `SQLite` vyřizuje vždy pouze jeden požadavek, samotný výkon databáze je relativně vysoký. Čtení dat má prioritu před zápisem. Proto se používá například pro CMS (content management system) weby či jako připojená databáze pro desktopové nebo mobilní aplikace.

Databázový engine čte a zapisuje data do souborů. Z funkcionalit klasických databázových serverů je tu podpora více tabulek. Indexování, podpora triggerů, můžete také vytvářet pohledy. `SQLite` podporuje transakce ACID(Atomicity, Consistency, Isolation, Durability). [1]

4.4.4 Aktivity Profil a Editace váhy

Vzhled aktivity **Profil** (Obr. 26) je složený z několika prvků typu **`ImageView`** pro zobrazení údajů o uživateli, které jsou opět načteny přes prvek `SharedPreferences`, do něhož byly údaje vloženy při registraci. Níže se nachází tlačítko pro odhlášení, který jednoduše využije `Intent` pro přeskočení do aktivity Přihlášení. Dole je pak tlačítko pro spuštění aktivity **Editace váhy**.

Obr. 26: Aktivita Profil

V Java kódu se nachází funkce *SectiXp()* (Obr. 27), která volá funkci *Sum()*, deklarovanou v **DatabaseHelper3** a přiřazuje za proměnnou ***sum***, součet všech zkušenostních bodů.

Funkce *VypocitejLevel()* pak následně podle proměnné ***sum*** stanovuje aktuální level uživatele.

```
public void SectiXp()
{
    sum = mojeDB.Sum();

    TVXp = (TextView) findViewById(R.id.tvProfilXp2);
    TVXp.setText(Integer.toString(sum));

}

public void VypocitejLevel()
{
    if (sum >= 100)
    {
        TVLevel = (TextView) findViewById(R.id.tvProfilLevel2);
        TVLevel.setText("1");
    }

    if (sum >= 200)
    {
        TVLevel = (TextView) findViewById(R.id.tvProfilLevel2);
        TVLevel.setText("2");
    }

    if (sum >= 300)
    {
        TVLevel = (TextView) findViewById(R.id.tvProfilLevel2);
        TVLevel.setText("3");
    }

    if (sum >= 400)
    {
        TVLevel = (TextView) findViewById(R.id.tvProfilLevel2);
        TVLevel.setText("4");
    }
```

Obr. 27: Aktivita Profil – Java

Vzhled aktivity **Editace váhy** (Obr.28) se skládá z několika prvků tipu **`ImageView`**, **`EditText`** a tlačítek pro zadání aktuální váhy nebo smazání vážení pomocí jeho id.

Design obsahuje také **`widget GraphView`**, který slouží pro zobrazení grafu. X-ová osa grafu zobrazuje id jednotlivých vážení a Y-ová osa údaj o váze v kilogramech. Tyto údaje jsou načítány z další `SQLite` databáze.

Pro použití grafů v Android aplikaci je nutné nejprve v podsložce **`build.gradle`**, která se nachází ve složce **`Gradle Scripts`**, přidávání grafů povolit. K tomu slouží červeně podtržený řádek na Obr. 29.

Obr. 28: Aktivita Editace Váhy

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.2.0'
    compile 'com.android.support:support-v4:25.2.0'
    compile 'com.android.support:design:25.2.0'
    compile 'com.jjoe64:graphview:4.2.1'
    testCompile 'junit:junit:4.12'
    compile 'com.google.android.gms:play-services-appindexing:8.4.0'
    compile 'pl.droidsonroids.gif:android-gif-drawable:1.1.+'
}
```

Obr. 29: Gradle Scripts

Tlačítko historie vážení otevírá `SQLite` databázi, která používá pomocnou aktivitu **DatabaseHelper2** a je oproti předešlému příkladu doplněná o funkci pro mazání položek z databáze. Kód musí být upraven jak v pomocné aktivitě **DatabaseHelper2** (Obr. 30) tak i v Aktivitě **Editace váhy** (Obr. 31).

```
public Integer deletedata(String id)
{
    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete("Vazeni_table", "ID = ?", new String[] {id});
}
```

Obr. 30: DatabaseHelper2

```
public void DeleteData()
{
    btnDelete.setOnClickListener(
            (v) → {

                    Integer deletedRows = myDb.deletedata(ETId.getText().toString());

                    if(deletedRows > 0)
                    {
                        Toast.makeText(Vaha.this, "Vaše vážení bylo smazáno.", Toast.LENGTH_LONG).show();
                    }
                    else
                    {
                        Toast.makeText(Vaha.this, "Vaše vážení nebylo smazáno.", Toast.LENGTH_LONG).show();
                    }
            }
    );
}
```

Obr. 31: DeleteData()

V Java kódu aktivity **Editace váhy** (Obr. 32) se situuje ovládání grafu. V řídící funkci *OnCreate()* dojde k inicializaci grafu. Tento kód se nachází také ve funkci pro přidání vážení, z důvodu, aby se graf po přidání nových hodnot aktualizoval.

```
GraphView graph = (GraphView) findViewById(R.id.graph);
LineGraphSeries<DataPoint> series = new LineGraphSeries<DataPoint>(getData());
graph.addSeries(series);
```

Obr. 32: Aktivita Editace Váhy Java – 1

Vykreslování bodů do grafu pak řídí funkce *getData()* (Obr. 33).

```
private DataPoint[] getData()
{
    String[] columns={"ID", "VAHA"};
    Cursor cursor = sqLiteDatabaze.query("Vazeni_table", columns, null, null, null, null, null);

    DataPoint[] dp=new DataPoint[cursor.getCount()];

    for(int i=0;i<cursor.getCount();i++)
    {
        cursor.moveToNext();
        dp[i]=new DataPoint(cursor.getInt(0),cursor.getInt(1));
    }
    return dp;
}
```

Obr. 33: Aktivita Editace váhy Java - 2

4.4.5 Aktivita Statistiky

Aktivita měla původně sloužit k zobrazení grafu historie vážení uživatele, ale z důvodu přehlednosti byl graf nakonec přesunut rovnou do aktivity **Editace váhy**.

Tato aktivita nabízí, ale i další eventuální využití, např. pro grafy zkušenostních bodů získaných za tréninky nebo počet tréninků odtrénovaných v jednotlivých měsících.

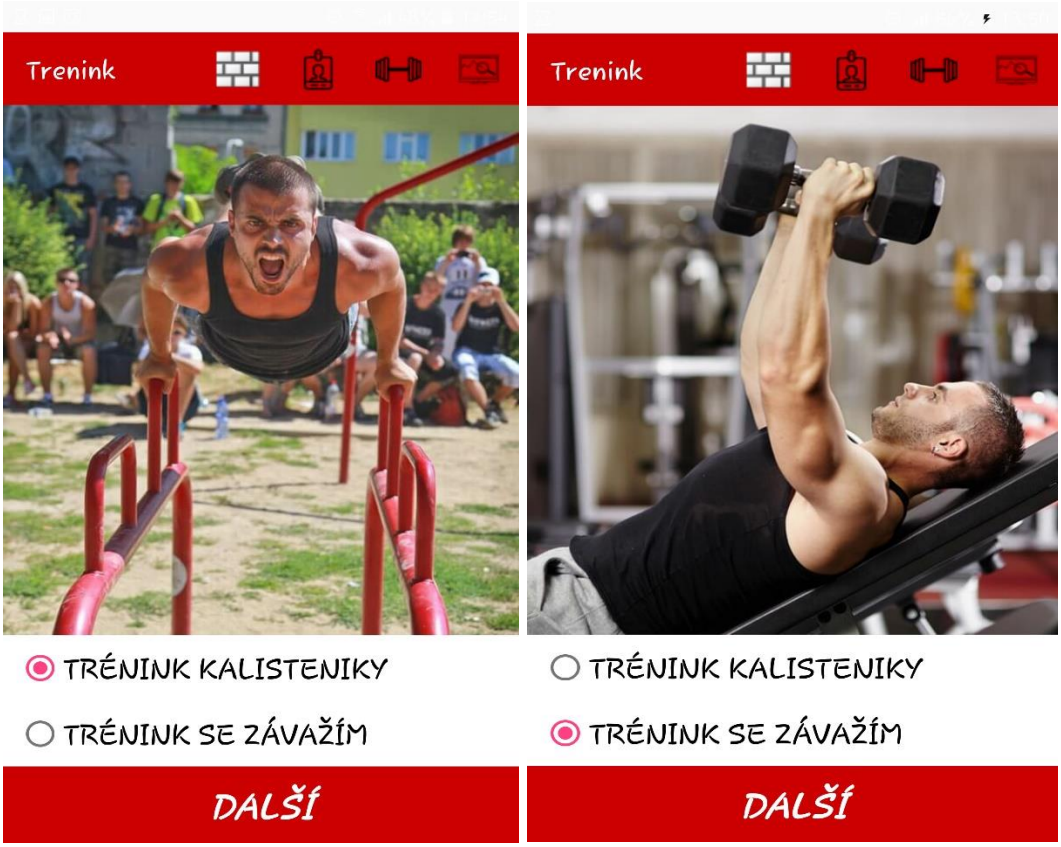
4.5 Aktivita Trénink

Tato aktivita je složena z několika dílčích aktivit a tvoří nejsložitější a nejdůležitější aktivitu celé aplikace.

4.5.1 Volba tréninku

V první řadě nabídne aplikace uživateli možnost zvolit si trénink se závažím nebo trénink s vlastní vahou těla tzv. kalistenika (Obr. 34).

Vzhled je tvořen dvěma prvky **`RadioButton`** umístěnými v **`RadioGroup`**, jejichž označení mění pozadí v **`FrameLayoutu`**.



Obr. 34: Volba tréninku 1

Při volbě tréninku se závažím se uživateli zobrazí možnost výběru buď tréninku cíleného na jednotlivé svalové partie nebo předem definovaných tzv. splitů (Obr. 35).

Tréninkový split je pravděpodobně nejrozšířenější styl tréninku a jedná se o trénink jedné velké svalové partie dohromady s jednou malou svalovou partií např. hrudník a triceps.

Obr. 35: Volba tréninku 2

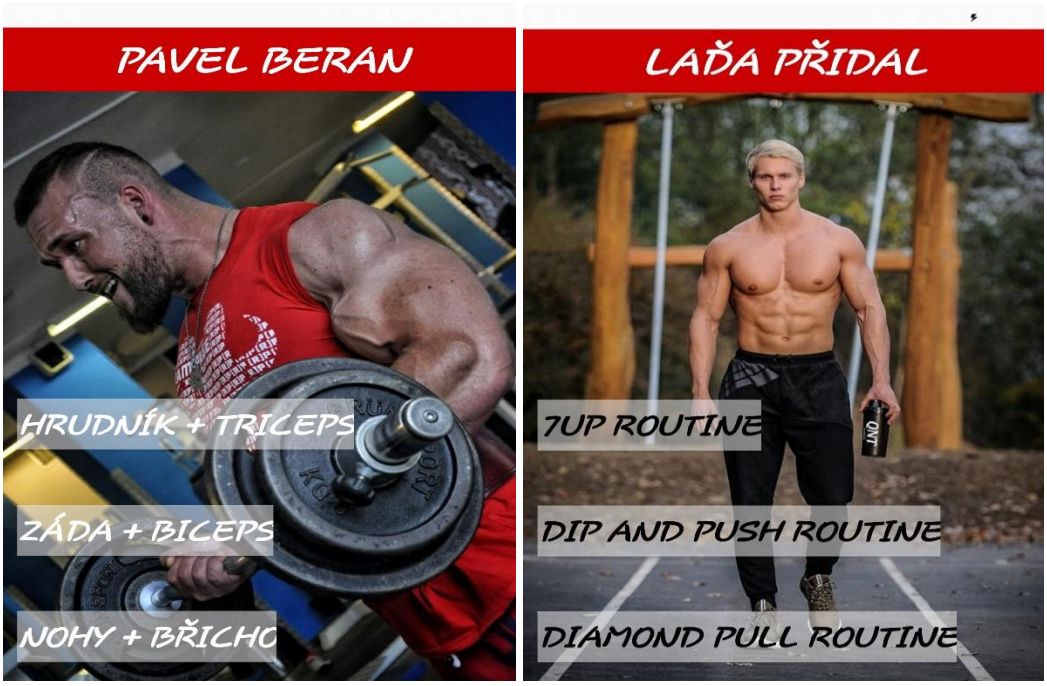
Při volbě tréninku kalisteniky (Obr. 36) možnost tréninku jednotlivých svalových partii odpadá z důvodu neschopnosti izolovat určité svalové partie tak efektivně jako právě při tréninku se závažím.

Obr. 36: Volba kalistenika

Při volbě předvolených tréninků, se zobrazí stručná charakteristika jednotlivých představitelů těchto stylů tréninků (Obr. 37).

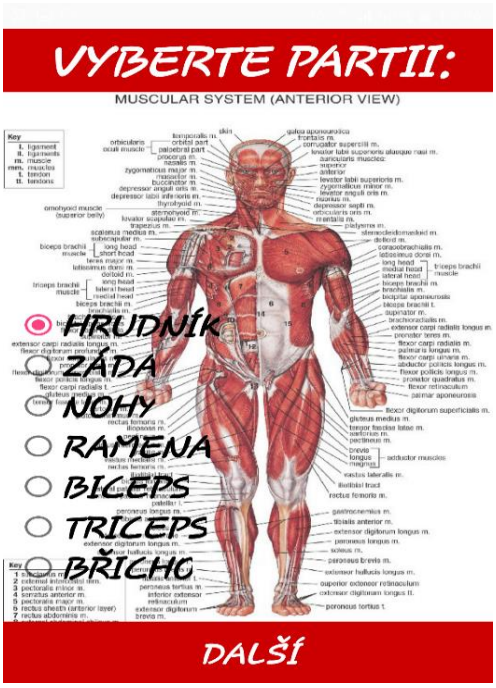
Obr. 37: Volba tréninku 3

Po stisknutí tlačítka Další se uživateli zobrazí aktivity pro výběr jednotlivých předdefinovaných setů (Obr. 38).



Obr. 38: Volba tréninku 4

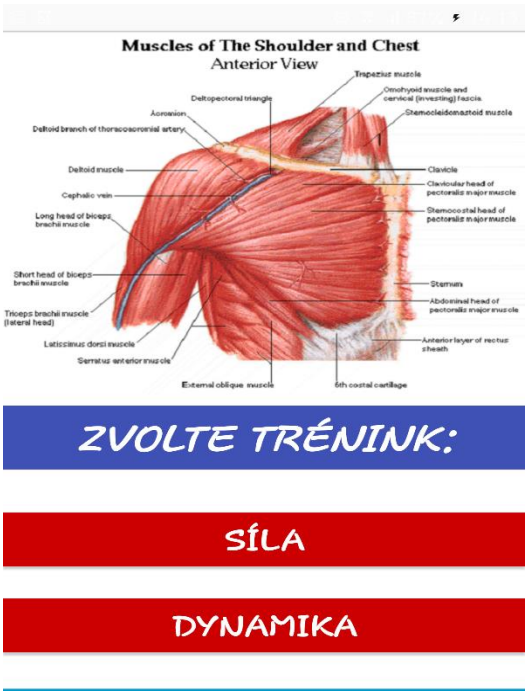
Při tréninku se závažím cíleného na jednotlivé svalové partie se zobrazí aktivita určená pro výběr partie (Obr. 38). Vzhled je opět tvořen **`RadioButtony`** uloženými v **`RadioGroup`**.



Obr. 39: Volba tréninku 5

Po stisku tlačítka Další se prvně zobrazí aktivita pro zvolení stylu tréninku. Aplikace dává na výběr trénink sestavěný buď pro zisk síly a svalového objemu nebo pro získání dynamiky (Obr. 40).

Po zvolení stylu tréninku se uživateli zobrazí aktivita určená pro volbu obtížnosti tréninku (Obr. 41). Aplikace dává uživateli na výběr ze tří stupňů náročnosti tréninku, které se odráží i na počtu získaných zkušenostních bodů.



Obr. 40: Volba tréninku 6

Obr. 41: Volba tréninku 7

Java kódy všech aktivit týkajících se volby tréninku jsou téměř identické, jde jen o vytvoření určité proměnné, specifikující trénink např. proměnná pro partii, obtížnost a styl. Všechny tyhle proměnné jsou postupně odesílány z jedné aktivity na druhou opět pomocí prvku `Intent` a všechny se sejdou v následující aktivitě: **Rozpis tréninku**.

4.5.2 Aktivita Rozpis tréninku

Design obsahuje horní lištu pro zobrazující aktuální aktivitu, čtyři **`ImageView`** pro zobrazení vybraných kategorií tréninku, prvek **`ScrollView`** pro vypsání jednotlivých cviků s hmotností závaží i počtem opakování (Obr. 42). **`ScrollView`** funguje stejně jako **`ImageView`** s tou změnou, že pokud je zobrazovaných dat více, než se vleze na display, lze prstem text popotáhnout.

Obr. 42: Aktivita Rozpis tréninku

V Java kódu (Obr. 43) musí aktivita nejdříve přijmout a definovat proměnné určující specifikaci tréninku, které jsou odeslané z předchozích aktivit pomocí prvku `Intent`.

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_rozpis_treninku);
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_NOSENSOR);

    TVZacPokPro = (TextView) findViewById(R.id.tvZacPokPro);
    intentObtiznost = getIntent();
    Obtiznost = intentObtiznost.getStringExtra("OBTIZNOSTTRENINKUKALI");
    TVZacPokPro.setText(Obtiznost);

    TVSilDynObo = (TextView) findViewById(R.id.tvSilDynObo);
    intentSilDynObo = getIntent();
    SilDynObo = intentSilDynObo.getStringExtra("KRDRUHTRENINKU2");
    TVSilDynObo.setText(SilDynObo);

    TVZavKali = (TextView) findViewById(R.id.tvZavKali);
    intentZNK = getIntent();
    ZavaziNeboKali = intentZNK.getStringExtra("ZavaziNeboKali");
    TVZavKali.setText(ZavaziNeboKali);

    TVParLadaBer = (TextView) findViewById(R.id.tvParLadaBer);
    intentJakaPartieKali = getIntent();
    JakaPartieKali = intentJakaPartieKali.getStringExtra("JakaPartieKali");
    TVParLadaBer.setText(JakaPartieKali);
```

Obr. 43: Aktivita Rozpis tréninku – Java 1

Poté co jsou definované proměnné specifikující trénink, se pomocí nich udělá logika, podle které se uživateli zobrazí určitá sada cviků (Obr. 43).

```
if (ZavaziNeboKali2.equals("Závaží"))
{
    if (JakaPartieKali2.equals("Ramena"))
    {

    }
    if (JakaPartieKali2.equals("Biceps"))
    {

    }
    if (JakaPartieKali2.equals("Triceps"))
    {

    }
    if (JakaPartieKali2.equals("Hrudník"))
    {
        if (SilDynObo2.equals("Síla"))
        {
            if (Obtiznost2.equals("Začátečník"))
            {
                TreninkZavaziHrudnikZacatecnik();
            }
            if (Obtiznost2.equals("Pokročilý"))
            {
                TreninkZavaziHrudnikPokrocily();
            }
            if (Obtiznost2.equals("Profesionál"))
            {
                TreninkZavaziHrudnikProfesional();
            }
        }
        if (SilDynObo2.equals("Dynamika"))
        {

    }
    if (JakaPartieKali2.equals("Záda"))
    {

    }
    if (JakaPartieKali2.equals("Břicho"))
    {

    }
    if (JakaPartieKali2.equals("Nohy"))
    {

    }
    if (JakaPartieKali2.equals("Pavel Beran"))
    {
        if(SilDynObo2.equals("Hrudník + Triceps"))
        {
            TreninkPBHrudnikBiceps();
        }

        if(SilDynObo2.equals("Záda + Biceps"))
        {

        }

        if(SilDynObo2.equals("Nohy + Břicho"))
        {

        }
    }
}
if (ZavaziNeboKali2.equals("Kalistenika"))
{
    if (JakaPartieKali2.equals("Laďa Přidal"))
    {


        if(SilDynObo2.equals("7Up Routine"))
        {
            SEVENUPROUTINE();
        }

        if(SilDynObo2.equals("Dip Push Routine"))
        {

        }

        if(SilDynObo2.equals("Diamond Pull Routine"))
        {

        }
    }
}
```

Obr. 44: Aktivita Rozpis tréninku – Java 2

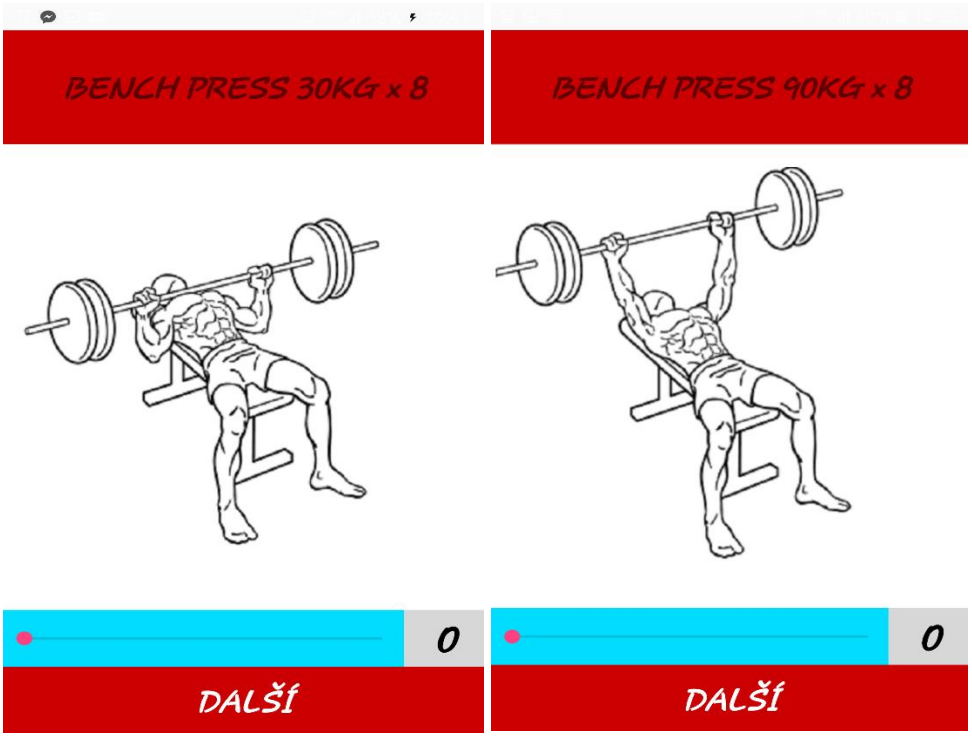
Funkce *TreninkZavaziHrudnikZacatecnik()* (Obr. 45) obsahuje údaj s názvem cviku, hmotností závaží a počtem opakování. Tyto údaje jsou zobrazeny do výše zmiňovaného **`ScrollView`** a také vloženy do `SharedPreferences` k dalšímu použití. Do funkce je vložena doplňující proměnná ***DelkaTreninku***, určující délku odpočítávané pauzy v následující aktivitě: **Chod tréninku 1** a proměnná ***PocetCviku***, která určuje, kdy trénink končí.

```
public void TreninkZavaziHrudnikZacatecnik()
{
    SharedPreferences sp2 = getSharedPreferences("CvikSerie", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor2 = sp2.edit();

    TVRozepsanePolozky = (TextView) findViewById(R.id.tv1);
    CvikSerie1 = "BENCH PRESS 30KG x 8";
    TVRozepsanePolozky.setText(CvikSerie1);
    editor2.putString("CvikSerie1", CvikSerie1);
    editor2.commit();

    TVRozepsanePolozky2 = (TextView) findViewById(R.id.tv2);
    CvikSerie2 = "BENCH PRESS 40KG x 6";
    TVRozepsanePolozky2.setText(CvikSerie2);
    editor2.putString("CvikSerie2", CvikSerie2);
    editor2.commit();

    TVRozepsanePolozky3 = (TextView) findViewById(R.id.tv3);
    CvikSerie3 = "BENCH PRESS 50KG x 4";
    TVRozepsanePolozky3.setText(CvikSerie3);
    editor2.putString("CvikSerie3", CvikSerie3);
    editor2.commit();

    TVRozepsanePolozky4 = (TextView) findViewById(R.id.tv4);
    CvikSerie4 = "BENCH PRESS 60KG x 2";
    TVRozepsanePolozky4.setText(CvikSerie4);
    editor2.putString("CvikSerie4", CvikSerie4);
    editor2.commit();

    TVRozepsanePolozky5 = (TextView) findViewById(R.id.tv5);
    CvikSerie5 = "BENCH PRESS 80KG x 1";
    TVRozepsanePolozky5.setText(CvikSerie5);
    editor2.putString("CvikSerie5", CvikSerie5);
    editor2.commit();

    DelkaTreninku = "kratky";
    PocetCviku = 5;
}
```

Obr. 45: Aktivita Rozpis tréninku – Java 3

4.5.3 Aktivity Chod tréninku 1 a 2

Vzhled této aktivity (Obr. 46) se skládá z horní lišty pro zobrazení údaje o cviku, pod ní se promítá animace daného cviku. V dolní části je pak **`SeekBar`** pro zadání počtu splněných opakování u daného cviku a **`ImageView`**, kde se údaj ze **`SeekBaru`** zobrazuje. Níže je pak tlačítko další, které spustí odpočet pauzy.



Obr. 46: Aktivita Chod tréninku 1

V Java kódu (Obr. 47) si aktivita opět bere pomocí prvku `Intent` proměnné, odeslané z předešlé aktivity a definuje je. Inicializuje **`SeekBar`** a **`ImageView`**, pro zobrazení údaje ze **`SeekBaru`**.

V kódu pro ovládání **`SeekBaru`** se podle proměnné ***DelkaTreninku*** a ***PocetCviku*** vytvoří logika programu pro uložení splněných opakování pro jednotlivé cviky a jejich odeslání do `SharedPreferences`.

```
intentPocetCviku = getIntent();
PocetCviku = intentPocetCviku.getIntExtra("PocetCviku", 0);

intentDelkaTreninku = getIntent();
DelkaTreninku = intentDelkaTreninku.getStringExtra("DelkaTreninku");

SB = (SeekBar) findViewById(R.id.seekBar);
SB.setMax(30);

TVUdelanaOpak = (TextView) findViewById(R.id.tvUdelanaOpak);
TVUdelanaOpak.setText("0");

SB.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
{
    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b)
    {
        TVUdelanaOpak.setText(String.valueOf(i));

        SharedPreferences sp = getSharedPreferences("UdelanaOpakovani", Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sp.edit();

        if (DelkaTreninku.equals("kratky"))
        {
            if (PocetCviku == 5) {
                UdelanaOpakPrvniSerie1 = i;


                editor.putInt("UdelanaOpakPrvniSerie1", UdelanaOpakPrvniSerie1);
                editor.commit();
            }
            if (PocetCviku == 4) {
                UdelanaOpakPrvniSerie2 = i;

                editor.putInt("UdelanaOpakPrvniSerie2", UdelanaOpakPrvniSerie2);
                editor.commit();

            }
            if (PocetCviku == 3) {
                UdelanaOpakPrvniSerie3 = i;

                editor.putInt("UdelanaOpakPrvniSerie3", UdelanaOpakPrvniSerie3);
                editor.commit();
            }
            if (PocetCviku == 2) {
                UdelanaOpakPrvniSerie4 = i;

                editor.putInt("UdelanaOpakPrvniSerie4", UdelanaOpakPrvniSerie4);
                editor.commit();
            }
            if (PocetCviku == 1) {
                UdelanaOpakPrvniSerie5 = i;

                editor.putInt("UdelanaOpakPrvniSerie5", UdelanaOpakPrvniSerie5);
                editor.commit();
            }
        }
```

Obr. 47: Aktivita Chod tréninku 1 – Java

Funkce *VypisCviku()* (Obr.48) slouží k zobrazení aktuálního cviku. Jednotlivé cviky byly nadefinovány a vloženy do `SharedPreferences` v minulé aktivitě a tato aktivita si cviky ze `SharedPreferences` vytáhne a zobrazí do horní lišty. Opět jsou zde použity proměnné ***DelkaTreninku*** a ***PocetCviku*** nutné pro chod programu.

```
public void VypisCviku()
{
    SharedPreferences sp2 = getSharedPreferences("CvikSerie", Context.MODE_PRIVATE);

    CvikSerie1 = sp2.getString("CvikSerie1", "");
    CvikSerie2 = sp2.getString("CvikSerie2", "");
    CvikSerie3 = sp2.getString("CvikSerie3", "");
    CvikSerie4 = sp2.getString("CvikSerie4", "");
    CvikSerie5 = sp2.getString("CvikSerie5", "");
    CvikSerie6 = sp2.getString("CvikSerie6", "");
    CvikSerie7 = sp2.getString("CvikSerie7", "");
    CvikSerie8 = sp2.getString("CvikSerie8", "");
    CvikSerie9 = sp2.getString("CvikSerie9", "");
    CvikSerie10 = sp2.getString("CvikSerie10", "");
    CvikSerie11 = sp2.getString("CvikSerie11", "");
    CvikSerie12 = sp2.getString("CvikSerie12", "");
    CvikSerie13 = sp2.getString("CvikSerie13", "");
    CvikSerie14 = sp2.getString("CvikSerie14", "");
    CvikSerie15 = sp2.getString("CvikSerie15", "");
    CvikSerie16 = sp2.getString("CvikSerie16", "");

    TVChodCvik = (TextView) findViewById(R.id.tvChodCvik);

    if (DelkaTreninku.equals("kratky"))
    {
        if (PocetCviku == 5)
        {
            TVChodCvik.setText(CvikSerie1);

        }
        if (PocetCviku == 4)
        {
            TVChodCvik.setText(CvikSerie2);

        }
        if (PocetCviku == 3)
        {
            TVChodCvik.setText(CvikSerie3);

        }
        if (PocetCviku == 2)
        {
            TVChodCvik.setText(CvikSerie4);

        }
        if (PocetCviku == 1)
        {
            TVChodCvik.setText(CvikSerie5);
        }
    }
}
```

Obr. 48: Aktivita Chod tréninku 1 – Java 2

Animace (Obr. 49) zobrazující aktuální cvik jsou řešeny jinak než v předchozích případech. Důvodem byla nestabilita programu.

Tento problém byl vyřešen vytvořením animace ve formátu tzv. GIF. Animace tedy neprobíhá v programu, ale je obsažena již v souboru, který je do programu implementován. Pro použití gifů v Android aplikaci je nutné nejprve v podsložce build.gradle, která se nachází ve složce Gradle Scripts přidávání grafů povolit. K čemuž slouží červeně podtržený řádek na Obr. 50.

Funkce si z horní lišty bere název zrovna probíhajícího cviku, ukládá ho do proměnné ***AnimaceCvik*** a podle klíčových slov jednotlivých názvů cviků nastavuje animaci.

```
public void AnimaceCviku()
{

    GifImageView GIWAnimCviku = (GifImageView) findViewById(R.id.giwAnimCviku);
    AnimaceCvik = TVChodCvik.getText().toString();


    if (AnimaceCvik.contains("BENCH"))
    {
        GIWAnimCviku.setBackgroundResource(R.drawable.gifbench);
    }

    if (AnimaceCvik.contains("ROZPAŽOVÁNÍ"))
    {
        GIWAnimCviku.setBackgroundResource(R.drawable.gifrozpaz);
    }

    if (AnimaceCvik.contains("TLAKY"))
    {
        GIWAnimCviku.setBackgroundResource(R.drawable.giztlaky);
    }


    if (AnimaceCvik.contains("KICK"))
    {
        GIWAnimCviku.setBackgroundResource(R.drawable.gifkbc);
    }

}
```

Obr. 49: Aktivita Chod Tréninku 1 – Java 3

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.2.0'
    compile 'com.android.support:support-v4:25.2.0'
    compile 'com.android.support:design:25.2.0'
    compile 'com.jjoe64:graphview:4.2.1'
    testCompile 'junit:junit:4.12'
    compile 'com.google.android.gms:play-services-appindexing:8.4.0'
    compile 'pl.droidsonroids.gif:android-gif-drawable:1.1.+'
}
```

Obr. 50: Gradle Scripts

Vzhled aktivity **Chod tréninku 2** (Obr. 51) je složen z horní lišty s textem odpočet pauzy, **`ImageView`** pro zobrazení odpočítávaného času a tlačítka pro přeskočení pauzy.

Obr. 51: Aktivita Chod tréninku 2

Aktivita přijímá přes `Intent` informaci o délce tréninku, podle které nastavuje délku pauzy mezi cviky (Obr. 52).

```
    intentDelkaTreninku = getIntent();
    DelkaTreninku = intentDelkaTreninku.getStringExtra("DelkaTreninku");

    Odpocitavej();
    OnClickButtonListener222();
}

private void Odpocitavej()
{
    CAS = (TextView) findViewById(R.id.cas);

    if (DelkaTreninku.equals("kratky"))
    {
        CAS.setText("120");
        CDT = new CountDownTimer(120 * 1000, 1000)
        {
            @Override
            public void onTick(long millisUtilFinished)
            {
                CAS.setText("" + millisUtilFinished / 1000);
            }

            @Override
            public void onFinish()
            {
                CAS.setText("HOTOVO!");

            }
        };
    }
```

Obr. 52: Aktivita Chod tréninku 2 – Java

Aktivita obsahuje i funkci *VyhodnotPreskoceni()* (Obr. 53), která si přes `Intent` bere informaci o počtu cviků, které jednotlivý trénink obsahuje. Tuto informaci obsahuje proměnná ***PocetCviku*** a program od ní odečte 1.

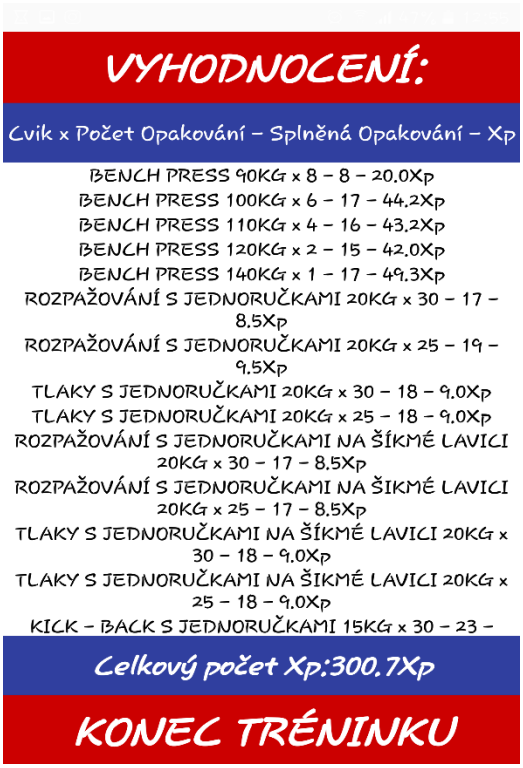
V případě, že uživatel ještě nesplnil všechny cviky, spouští se aktivita **Chod tréninku 1**, a v případě, že již jsou všechny cviky splněné, se spouští aktivita **Vyhodnocení tréninku**.

```
public void VyhodnotPreskoceni()
{
    intentDelkaTreninku = getIntent();
    DelkaTreninku = intentDelkaTreninku.getStringExtra("DelkaTreninku");

    intentPocetCviku = getIntent();
    PocetCviku = intentPocetCviku.getIntExtra("PocetCviku", 0) - 1;

    if(PocetCviku != 0)
    {
        Intent intent = new Intent("com.example.martin.bpostre.ChodTreninku1");

        intent.putExtra("PocetCviku", PocetCviku);
        intent.putExtra("DelkaTreninku", DelkaTreninku);

        startActivity(intent);
    }

    if(PocetCviku == 0)
    {
        Intent intent = new Intent("com.example.martin.bpostre.VyhodnoceniTreninku");

        intent.putExtra("DelkaTreninku", DelkaTreninku);

        startActivity(intent);
    }

}
```

Obr. 53: Aktivita Chod tréninku 2 – Java 2

4.5.4 Aktivita Vyhodnocení tréninku

Vzhled této aktivity (Obr. 54) je složen z horní lišty s textem vyhodnocení. Níže je umístěna lišta, která pojmenovává údaje zobrazené níže. Výsledky jsou zobrazovány ve formátu: cvik × počet opakování – splněná opakování – zkušenostní body do **`ScrollView`**. V dolní části je pak lišta, která zobrazí celkový počet zkušenostních bodů, které uživatel za celý trénink získal. Trénink se pak ukončí tlačítkem s textem Konec tréninku, který spustí aktivitu Zeď.



Obr. 54: Aktivita Vyhodnocení tréninku

V Java kódu této aktivity (Obr. 55) nejprve dojde k přijetí a deklaraci proměnné určující délku tréninku.

```
intentDelkaTreninku = getIntent();
DelkaTreninku = intentDelkaTreninku.getStringExtra("DelkaTreninku");
```

Obr. 55: Aktivita Vyhodnocení tréninku – Java 1

Ve Funkci *VypocetAVypisXp()* (Obr. 56) dojde k načtení údajů o specifikaci tréninku, názvů jednotlivých cviků a o splněných opakováních.

Program pak stanoví zkušenostní body, které uživatel během tréninku nasbíral a definuje proměnné obsahující údaje o aktuálním datu.

```
public void VypocetAVypisXp()
{
    if(DelkaTreninku.equals("kratky"))
    {
        SharedPreferences sp = getSharedPreferences("UdelanaOpakovani", Context.MODE_PRIVATE);

        UdelanaOpakPrvniSerie1 = sp.getInt("UdelanaOpakPrvniSerie1", 0);
        UdelanaOpakPrvniSerie2 = sp.getInt("UdelanaOpakPrvniSerie2", 0);
        UdelanaOpakPrvniSerie3 = sp.getInt("UdelanaOpakPrvniSerie3", 0);
        UdelanaOpakPrvniSerie4 = sp.getInt("UdelanaOpakPrvniSerie4", 0);
        UdelanaOpakPrvniSerie5 = sp.getInt("UdelanaOpakPrvniSerie5", 0);

        SharedPreferences sp7777 = getSharedPreferences("ObtiznostAStyl", Context.MODE_PRIVATE);

        Styl = sp7777.getString("Styl7777", "");
        Obtiznost = sp7777.getString("Obtiznost7777", "");
        ZavaziNeboKali = sp7777.getString("ZavaziNeboKali7777", "");
        JakaPartieKali = sp7777.getString("JakaPartieKali7777", "");

        SharedPreferences sp2 = getSharedPreferences("CvikSerie", Context.MODE_PRIVATE);

        CvikSerie1 = sp2.getString("CvikSerie1", "");
        CvikSerie2 = sp2.getString("CvikSerie2", "");
        CvikSerie3 = sp2.getString("CvikSerie3", "");
        CvikSerie4 = sp2.getString("CvikSerie4", "");
        CvikSerie5 = sp2.getString("CvikSerie5", "");

        if (Styl.equals("Síla"))
        {
            if (Obtiznost.equals("Začátečník"))
            {
                XP111 = UdelanaOpakPrvniSerie1 * 2;
                XP1 = Math.round(XP111 * 100) / 100.0;

                XP22 = UdelanaOpakPrvniSerie2 * 2.1;
                XP2 = Math.round(XP22 * 100) / 100.0;

                XP33 = UdelanaOpakPrvniSerie3 * 2.2;
                XP3 = Math.round(XP33 * 100) / 100.0;

                XP44 = UdelanaOpakPrvniSerie4 * 2.3;
                XP4 = Math.round(XP44 * 100) / 100.0;

                XP55 = UdelanaOpakPrvniSerie5 * 2.4;
                XP5 = Math.round(XP55 * 100) / 100.0;


                XPCelkove2 = XP111 + XP22 + XP33 + XP44 + XP55;
                XPCelkove = Math.round(XPCelkove2 * 100) / 100.0;

                c = Calendar.getInstance();

                DatumTreninku = c.get(Calendar.DATE);
                DatumTreninku2 = c.get(Calendar.MONTH);
                DatumTreninku3 = c.get(Calendar.YEAR);
            }
```

Obr. 56: Aktivita Vyhodnocení tréninku – Java 2

Po kliknutí na tlačítka Konec Tréninku aktivita přes prvek `Intent` spouští aktivitu **Zed'** a odesílá do ní všechny údaje o tréninku (Obr. 57).

```
public void OnClickButtonListener7777()
{
    BTNKonecTreninku = (Button) findViewById(R.id.btnVyhodnoceniKonec);
    BTNKonecTreninku.setOnClickListener(
            new View.OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    Intent intent = new Intent("com.example.martin.bpostre.Zed");

                    intent.putExtra("DatumTreninku", DatumTreninku);
                    intent.putExtra("DatumTreninku2", DatumTreninku2);
                    intent.putExtra("DatumTreninku3", DatumTreninku3);
                    intent.putExtra("Zkusenosti", XPCelkove);
                    intent.putExtra("Styl7777", Styl);
                    intent.putExtra("Obtiznost7777", Obtiznost);
                    intent.putExtra("ZavaziNeboKali7777", ZavaziNeboKali);
                    intent.putExtra("JakaPartieKali7777", JakaPartieKali);
                    startActivity(intent);
                }
            }
    );
}
```

Obr. 57: Aktivita Vyhodnocení tréninku – Java 3

5 ZÁVĚR

Sportovní aplikace, která je popsána v této práci, byla úspěšně odzkoušena v praxi. Aplikaci jsem osobně testoval při svém vlastním tréninku, a všechny funkce, které aplikaci tvoří, fungují zcela bez problému.

Jako další eventuální rozšíření programu přichází v úvahu vytvoření celistvých tréninkových programů např. po dobu měsíce. Splněné i ještě nesplněné tréninky by se zobrazovaly v kalendáři, který se nachází v aktivitě **Zed'** a vznikla by nová aktivita s databází, která by sloužila pro údaje o tréninkových programech.

Celá aplikace by mohla sloužit i jako sociální síť. Uživatel by měl např. na výběr vytvoření fotografie z tréninku. Fotografie by byla doplněna například o celkové dosažené zkušenostní body z tréninku a ukládala by se na uživatelův profil, ale i do jakési společné zdi všech uživatelů. Všichni uživatelé by si tak mohli navzájem ohodnotit své tréninky. Popřípadě by vznikli žebříčky na určité tréninky a uživatelé by mohli navzájem soupeřit, kdo získá větší počet zkušenostních bodů.

Program by mohla být rozšířen ještě o hudbu, jak pro pomalejší melodie po čas, kdy se uživatel nachází v menu, tak pro rychlejší po dobu, kdy uživatel má zapnutý trénink.

6 SEZNAM POUŽITÉ LITERATURY

[1] Lacko, Luboš. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[2] D'Souza, Ryan. *7 Incredible Features of Android Nougat That Make It A Worthy Treat*., [online]..[cit.21.4.2017]. Dostupné z: http://www.couponraja.in/theroyale/android-nougat-features/

[3] Android Studio: An IDE built for Android [online]..[cit.21.4.2017]. Dostupné z: https://cs.wikipedia.org/wiki/Android_Studio

[4] Perlík, Jan. *Jak se vyvíjel operační systém Android? Napříč jeho historií až do současnosti 2. díl*, [online]..[cit.21.4.2017]. Dostupné z: http://androidmarket.cz/android/jak-se-vyvijel-operacni-system-android-napric-jeho-historii-az-do-soucasnosti-2-dil/

[5] Evans, Jon. *Why is Android Studio still such a gruesome embarrassment?*, [online]..[cit.21.4.2017]. Dostupné z: https://techcrunch.com/2017/02/19/why-is-android-studio-still-such-a-gruesome-embarrassment/

7 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK

Layout

Layout slouží ke grafickému rozvržení vzhledu jednotlivých aktivit aplikace. Je to elegantní a flexibilní způsob jak uspořádat widgety. Jedná se tedy o plochu, do které se widgety umisťují. Existuje několik typů Layoutů např. FrameLayout nebo LinearLayout, vhodné pro různé typy aktivit.

Widget

Widget je dílčí prvek neboli komponent každé aplikace. Jedná se o miniaturní zobrazení aplikace, které můžou být vloženy do jiných aplikací např. do domovské stránky a dostávají pravidelné aktualizace. Widget je slovo nadřazené pro všechny komponenty popisované v této práci jako RadioButton, RadioGroub, ScrollView atd.

ImageView

Tento prvek se používá pro zobrazení libovolného obrázku, např. ikony. ImageView může načítat obrázky z různých zdrojů např. ze zdroje nebo od poskytovatelů obsahů. Postará se o výpočet rozlišení obrázku, tak aby mohl být použit v jakémkoliv rozvržení a poskytuje různé možnosti zobrazení, jako je změna velikosti a tónování.

EditText

EditText je prvek, který se používá pro získání dat od uživatele. V aplikaci, kterou tato práce popisuje, jsou prvky typu EditText použity např. pro získání přihlašovacích údajů nebo údajů o aktuálním stavu tělesné váhy.

RadioButton a RadioGroup

RadioButton je v podstatě jen kulaté zaškrtávací tlačítko, které nabývá dvou hodnot: hodnoty pro zapnuté a vypnuté tlačítko. Při stisknutí tlačítka, RadioButton nabude zapnuté hodnoty, a uprostřed kulatého tlačítka se objeví malá tečka. Při opakovaném stisknutí, RadioButton nabude opačné hodnoty a kulaté tlačítko zůstane prázdné.

Nejčastější způsob použití těchto widgetů je sdružit je do prvku RadioGroup. Toto zařazení způsobí, že jde současně zaškrtnout jen jeden RadioButton.

8 SEZNAM PŘÍLOH

PŘÍLOHY

Součástí bakalářské práce je CD obsahující program aplikace.